NAS2-11530

Convergence and Stability of Iterative Queueing Network Models

Subhash C. Agrawal
&
Peter J. Denning

July 1984
Revised November 1984

Research Institute for Advanced Computer Science

RIACS TR 84.7

(NASA-CR-187294) CONVERGENCE AND STABILITY OF INTERATIVE QUEUEING NETWORK MODELS (Research Inst. for Advanced Computer Science) 39 p

N90-71374

Unclas
00/62 0295375

Research Institute for Advanced Computer Science

Convergence and Stability of Iterative Queueing Network Models

Subhash C. Agrawal
BGS Systems, Inc.
&
Peter J. Denning
Research Institute for Advanced Computer Science

RIACS TR 84.7

July 1984
Revised November 1984

A class of iterative solutions to queueing network models is analyzed for stability and convergence. We prove that, when the iteration function is monotone increasing in its argument, there will be at least one convergent solution. This theorem is used to demonstrate the convergence of three iterative models from the literature: the Jacobson-Lazowska surrogate server model, the Bard-Schweitzer mean value model, and the Sevcik shadow CPU model. The first two models have a unique solution. The shadow CPU model may exhibit multiple solutions or, if one server has a superlinear rate function, may exhibit no solution. We conjecture that the monotonicity of the shadow-CPU iteration function of any queueing network is guaranteed when all servers in the network are sublinear. These behaviors are illustrated with simulation results.

Work reported herein was supported in part by Contract NAS2-11530 from the National Aeronautics and Space Administration (NASA) to the Universities Space Research Association (USRA) and in part by NSF Grant MCS81-01729 at Purdue University.

RIACS is located at the NASA Ames Research Center at Moffett Field, California.

Authors' present addresses:
Subhash C. Agrawal, BGS Systems, Inc., One University Office Park, Waltham, MA 02254.
Peter J. Denning, RIACS, Mail Stop 230-5, NASA Ames Research Center, Moffett Field, CA 94035 (ARPANET address: denning@riacs).

Convergence and Stability of Iterative Queueing Network Models

Subhash C. Agrawal
BGS Systems, Inc.
&
Peter J. Denning
Research Institute for Advanced Computer Science

July 1984
Revised November 1984

1. INTRODUCTION

Many computer systems exhibit behaviors so inhomogeneous that no direct product form queueing network model can accurately estimate throughput and response time. Examples are simultaneous resource possession, preemptive priorities, serialization on software locks, and blocking on full buffers.

To deal with these cases, performance analysts have been studying how to represent inhomogeneous behavior with special, possibly nonphysical, servers in a product form model. The parameters of the new servers are unknown and may be calculated by iteratively refining guesses; product form algorithms are used to obtain fast solutions of the model at each cycle of the iteration. On convergence, the performance metrics of the final product form solution are transformed to solutions of the original system.

An example is the "surrogate server" method devised by Jacobson and Lazowska [JACO82]. They used delay servers to model the extra queueing caused by jobs waiting for service from a secondary resource while holding a primary resource. Another example is the "shadow CPU" method of Sevcik [SEVC77]. He split a CPU serving high- and low-priority jobs into two, with the low-priority CPU's service time a degraded value of the original CPU's low-priority service time. In both these cases, guesses of the parameters of the extra servers were successively improved by using the solution of the modified, product form queueing model at each iteration step.

Iteration arises in other ways as well. The Bard-Schweitzer approximation to the mean-value equations for product form networks are solved by calculating a series of guesses of the device queue lengths [BARD79, SCHW79].

These and other iteration models have left several fundamental questions unanswered. Is there a general principle that explains when an iterative solution will arise? Will the iteration converge? If so, is the solution unique? Are there physical interpretations of multiple solutions?

This paper explores these questions. Section 2 defines a modeling schema in which iteration arises; this schema covers a wide variety of practical cases. Section 3 presents conditions on the iteration function that guarantee convergent solutions. Section 4 applies this result to obtain short proofs of convergence for the Jacobson-Lazowska, Sevcik, and Bard-Schweitzer models. Section 5 shows that the shadow CPU model may have multiple solutions. A simulation model verifies the physical significance multiple solutions. A system with a "superlinear" server (not likely to be encountered in practice) may have no solution at all.

1.1. Related Work

Several other authors have recently considered the convergence of iterative algorithms for queueing network models. The comprehensive paper by de Sousa e Silva *et al.* considers iterations arising in the class of "device-complement" models [deSO83]. In this class, a network is modeled by a collection of subnetworks, each consisting of one of the original servers and an equivalent server representing the rest of the original network. The parameters of the equivalent server in one subnetwork are derived from the performance measures of the other subnetworks. Starting from initial guesses of the performance measures, each of the subnetworks is solved again in turn until some convergence criterion is attained. The authors formulate the iteration as a nonlinear fixed-point equation and state that standard techniques of numerical analysis can be applied in specific cases to determine whether there is a unique, stable solution. They also present a new iterative solution for multiple-class networks.

Eager and Sevcik [EAGE83] and Agrawal [AGRA83] independently discovered proofs that the Bard-Schweitzer approximation converges for given initial conditions.

Galler and Bos derived an iterative approximation for the performance measures of a single-class database system in which transactions can block one another [GALL83]. They formulated the convergence question as a fixed-point problem and showed that their method produces unique solutions under realistic conditions.

Agrawal proposed a unified framework for the modeling processes associated with queueing networks [AGRA83]. He showed that fixed-point equations arise frequently in these processes and developed techniques for proving convergence of the solutions. This paper is based largely on that work.

2. AN ITERATIVE SCHEMA

An analysis of a computer system usually starts with an initial model, M_0, whose solution exists and is known to be sufficiently accurate. But because the equations of M_0 are typically too expensive for a direct solution, the analyst usually transforms M_0 into a simpler model, M, whose inexpensive solution approximates that of M_0.

For example, M_0 can denote the global balance equations over the system's state space and M can be a product form queueing network model. Direct solution of M_0 is normally too expensive or infeasible whereas direct solution of M is normally cheap and efficient. The accuracy of M's approximation to M_0's solution depends on the extent to which M_0 satisfies network homogeneity, the property that the flow rate between a pair of servers depends only on the queue length at the source.

Figure 1 illustrates these ideas. The original model, M_0, is mapped by a forward transformation, F, into a simpler model, M, whose solution is mapped backward as an approximation by the reverse transformation, R. The equations arising from this diagram have the form

$$
\begin{aligned}
P &= F(P_0, Q_0) \\
Q &= \mathrm{SOLVE}(M, P) \\
Q_0 &= R(Q)\,.
\end{aligned}
\tag{1}
$$

where

P_0, P	-	Parameters of M_0 and M, respectively
Q_0, Q	-	Solutions of M_0 and M, respectively
SOLVE(M, P)	-	An algorithm that calculates the solution of M given parameter values P.

This schema is very general. It says nothing specific about the F, R, or SOLVE mappings. The equations arising in specific cases can be straightforward (as later examples in this paper will show) or quite complicated (as in the Galler-Bos model [GALL83]). The solutions denoted by Q_0 and Q will be some subset of standard performance metrics such as utilizations, throughputs, response times, queue lengths, and steady-state probabilities.

Let us illustrate the schema of Figure 1 with two examples. The first is the product-form queueing network model. In this case, model M_0 is a system of flow balance equations over the system's states and model M is a system of flow-balance equations whose solution has the product form. The forward mapping, F, determines the values of one-step mean service times and visit ratios, S_i and V_i, according to the network homogeneity assumption. The solver, SOLVE(M, P), denotes any one of the queueing network solution algorithms, such as normalizing constant analysis or mean value analysis, that evaluates the equations in terms of the parameters S_i and V_i. The reverse map, R, is an identity.

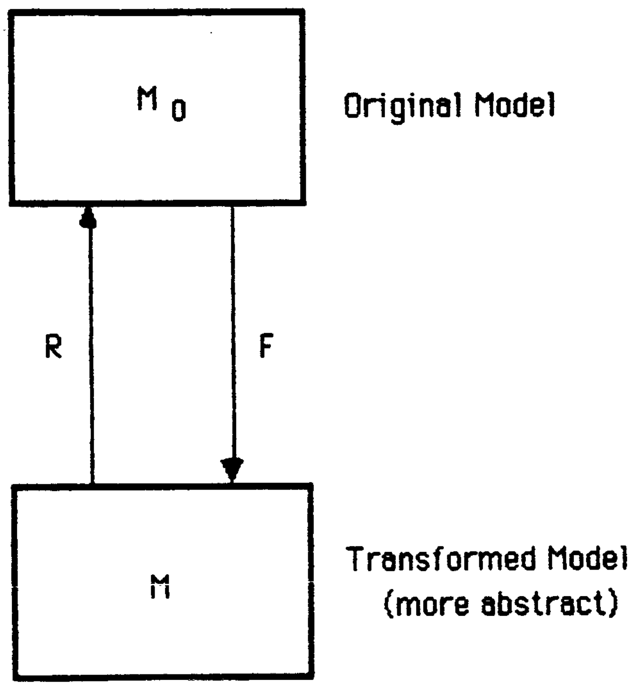


FIGURE 1. Basic Modeling Process.

The second example is Sevcik's shadow CPU model. Figure 2(a) shows the original system, in which high-priority (H) jobs and low-priority (L) jobs use the same CPU; H jobs preempt L jobs at the CPU. Figure 2(b) shows Sevcik's transformation, which splits the original CPU into two, one for H jobs and the other for L. The service time of CPU-H is the same as in the original system because H jobs never wait for L jobs. All the other parameters of the network are the same as in the original system. However, the service time of CPU-L is degraded from the original CPU's service time for L jobs, S_L , to reflect the effect of preemption:

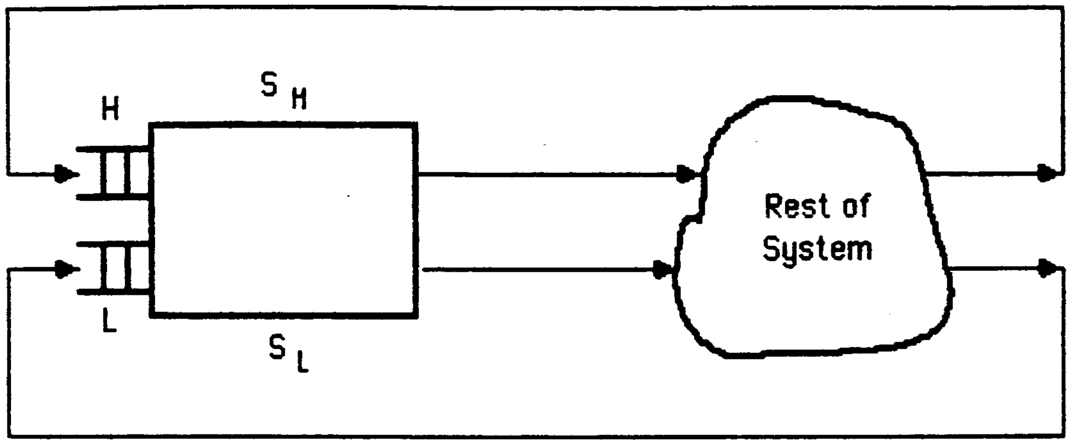
$$S_L{}' = \frac{S_L}{1-U_H} , \tag{2}$$

where U_H is the utilization of the original CPU by H jobs. Note how the forward mapping for $S_L{}'$ depends on both a parameter (S_L) and a metric (U_H) of the original model.

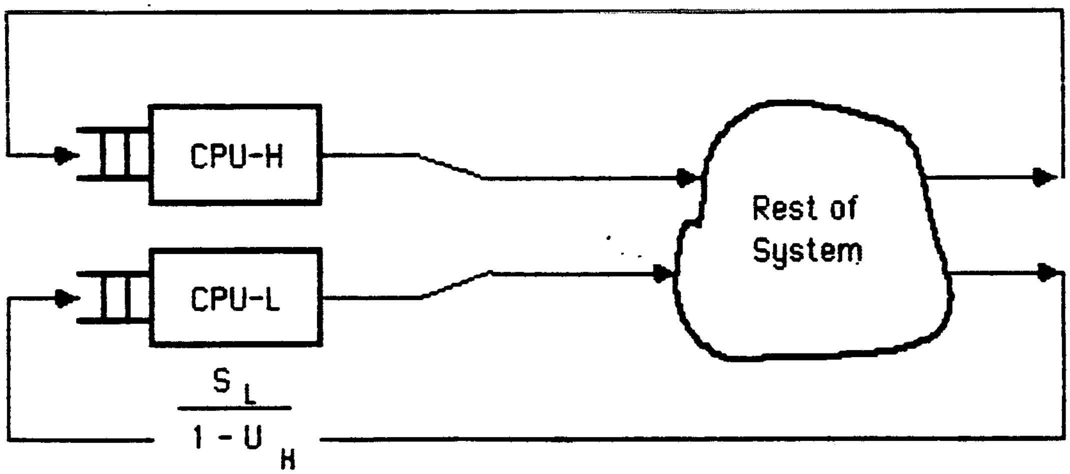
Because the utilization U_H is initially unknown, Sevcik's algorithm solves the split-CPU model iteratively to construct a series of trial utilizations, $U_H^{(0)}$, $U_H^{(1)}$, $\cdots$ in search of a convergent solution. Since U_H is the only unknown in the forward mapping, a complete solution of Equations (1) converges when U_H converges.

The Shadow CPU example illustrates a common aspect of modeling: components of the solution of one model can become components of the parameters of another model. The notation of Equation (1) is purposely general to make this clear.

In general, when the parameters of M depend on the unknown performance metrics Q_0, iteration can be used to refine successive guesses of Q_0 into a solution. When a single unknown metric, x of Q_0, is required to compute the parameters of M, the schema is:



(a) Model M_0



(b) Model M

FIGURE 2. Sevcik's Shadow CPU Model.

1. **Initialize:** $x^* := x^{(0)}$

2. **Repeat** $\{$ $x := x^*$

$$P := F(P_0, Q_0 - \{x\}, x)$$

$$Q := \text{SOLVE}(\text{M}, P) \tag{3}$$

$$Q_0 := R(Q)\ \}$$

until $|x - x^*| < \epsilon$

3. **Output** Q_0

This procedure generates a sequence of estimates $Q^{(0)}$, $Q^{(1)}$, $Q^{(2)}$,... until an error measure between successive estimates of the unknown metric is sufficiently small.

In the most abstract terms, this iteration schema is attempting a solution of the nonlinear fixed-point equation

$$x = I(x) \tag{4}$$

where I is an iteration function:

$$I(x) = R_x(\text{SOLVE}(F(P_0, Q_0 - \{x\}, x)))$$

and $R_x(Q)$ denotes the projection of x from set Q.

The same schema can be generalized to iterations involving two or more unknown performance metrics. We will not consider this further here.

As part of the construction of a model according to the schema of Figure 1, the analyst must ask:

1. For the given F, R, and SOLVE, does the iteration function I have any solution?

2. If so, will the Algorithm (3) converge?

3. Is the solution unique?

The next section characterizes the answers to these questions for a large class of practical systems.

3. CONDITIONS FOR CONVERGENCE

There are two basic approaches to proving that an iterative algorithm converges to a solution. The first is a direct application of the definition of convergence: showing that the sequence of estimates resulting from a given initial condition are successively closer to a given limit point. This idea underlies Theorem 1, below. The second is indirect: showing that the iteration function has properties sufficient to force any sequence of estimates resulting from an allowable initial condition to close on a limit point. This idea underlies Theorem 2, below. When applicable, the second approach is the more powerful because it relates the given F, SOLVE, and R mappings to the convergence question. The goal of this section is a general characterization of the second approach under realistic conditions.

These two basic approaches are stated below as theorems. They apply to a wide range of practical problems because performance metrics are often monotonic and bounded -- for example, the throughput is usually a strictly increasing function of any device's utilization [SURI83] and is bounded between 0 and a constant determined by the bottleneck device's maximum service rate [DENN78]. In the following discussion, we seek a solution to $x = I(x)$ in the range $[L, U]$ where $L < U$.

Theorem 1. Monotone Bounded Sequence Theorem.
Suppose the algorithm generates estimates x_0, x_1, $\cdots$ such that $x_i \leqslant U$ and, for $i > k$, $x_{i+1} \geqslant x_i$. Then the algorithm converges.

Sketch of Proof: If $x_{i+1} = x_i$ for $i > k$, the algorithm obviously converges. Assume $x_{i+1} > x_i$, let $d_i = U - x_i$, and note that $d_i > 0$ and $d_i > d_{i+1}$. Then

$$\lim_{j \to \infty} \frac{d_j}{d_i} = \lim_{j \to \infty} \frac{d_{i+1}}{d_i} \frac{d_{i+2}}{d_{i+1}} \cdots \frac{d_j}{d_{j-1}} = 0 ,$$

which implies the algorithm converges.

Corollary: If $x_i \geqslant L$ and $x_{i+1} \leqslant x_i$ for all $i > k$, the iteration converges.

Theorem 2 is based on a classical result from numerical analysis, which gives a sufficient condition for a convergent iteration function [STOE80]. A solution z to $x = I(x)$ is called a "fixed point" because the sequence of estimates of z will be fixed (at z) if ever one of the estimates equals z. A solution z is stable if the successor of each estimate is closer to z (i.e., $| I(x_i) - z | < | x_i - z |$); otherwise the solution is unstable. An unstable fixed point cannot be found by iterative algorithms because the successive estimates will diverge from it. An unstable fixed point cannot be observed in a physical system outside of a short observation interval because any small change will cause the system's state to drift away from that operating point.

Figure 3 illustrates the basic result from numerical analysis. If the solution z to $x = I(x)$ lies in the interval $[L, U]$ and the magnitude of the derivative, $|I'(x)|$, is less than 1 in that interval, the iteration will converge to z. In the figure, an estimate $x_i < z$ will generate a next estimate $x_{i+1} = I(x_i)$ closer to z. (A similar statement holds for an estimate $x_i > z$.) If the iteration function is not monotone increasing but $|I'(x)| < 1$, x_{i+1} may be on the opposite side of z from x_i, but will still be closer.

While the iteration functions for most queueing network models are bounded mootone, and continuous, they are seldom well enough behaved for this basic result to apply [AGRA83, SURI83]. In general, the entire range of x-values will be partitioned into subranges that alternate between $I(x)$ having slope less than 1 and slope greater than 1. If $y = I(x)$ crosses $y = x$ in a subrange of slope less than 1, that crossing will be a stable fixed point; other crossings will be unstable.

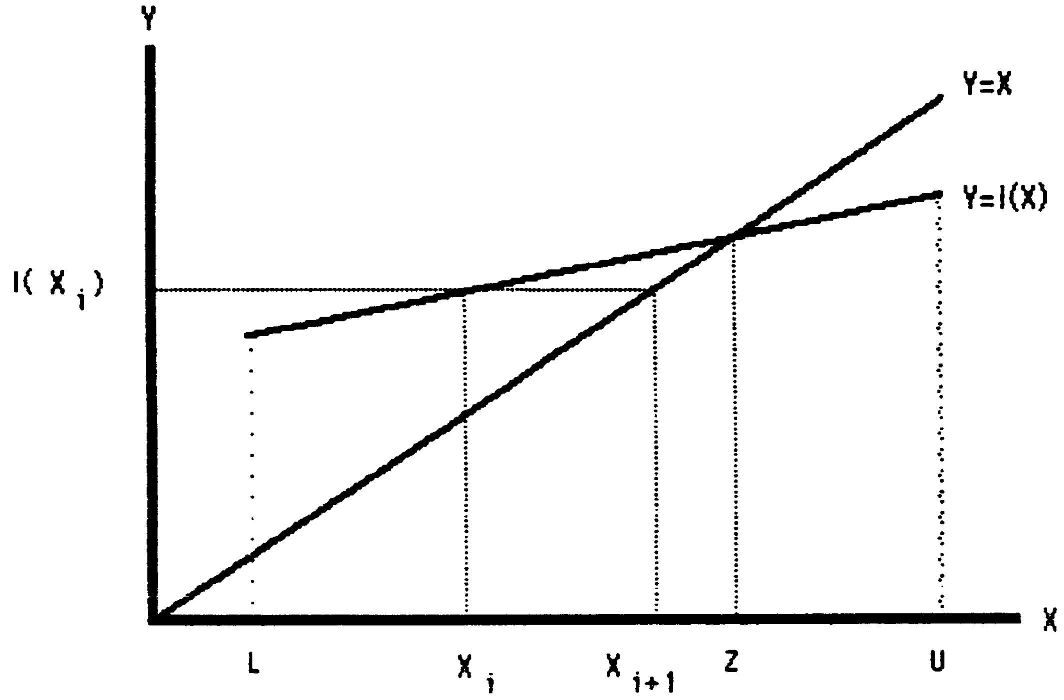


FIGURE 3. Basic Numerical Result.

Theorem 2. Monotone Bounded Function Theorem.

Suppose that $I(x)$ is monotone increasing, bounded, and continuous on the interval $[L, U]$. Suppose $I(L) > L$ and $I(U) < U$. Then:

1. There is at least one stable fixed point in $[L, U]$ and

2. Stable and unstable fixed points alternate.

Sketch of Proof. Figure 4 illustrates that a continuous bounded curve that lies above the 45-degree line for $x = L$ and below that line for $x = U$ must cross that line an odd number of times. A crossing may be called a down (up) crossing if $I(x)$ passes from above the line to below (or below to above) as x increases. Down- and up-crossings must alternate. In Figure 4, z_1, z_3, and z_5 are down-crossings while z_2 and z_4 are up-crossings.

The slope of the monotone increasing function $I(x)$ is everywhere at least 0. Near a down-crossing, $I'(x)$ must be less than 1, the slope of the 45-degree line. Near an up-crossing, $I'(x)$ must be greater than 1. The basic numerical result then implies that the down-crossings are stable fixed points and the up-crossings are unstable.

The theorem holds even when $I(x)$ is tangent to $y = x$; in this case a point of tangency will be both a stable fixed point and an unstable fixed point, depending on the direction of approach.

Corollary. Theorem 2 also holds when $I(L) = L$ or $I(U) = U$.

Sketch of Proof. If $I(L) = L$, then $z_1 = L$. If z_1 is a down-crossing, Theorem 2 obviously holds; if z_1 is an up-crossing, there must be a down-crossing $z_2 \leqslant U$ on the path from $I(z_1^+)$ to $I(U)$. Similarly, if $I(U) = U$, then $z_k = U$. If z_k is a down-crossing, Theorem 2 obviously holds; if z_k is an up-crossing, there must be a down-crossing $z_{k-1} \geqslant L$ on the path from $I(L)$ to $I(z_k^-)$.

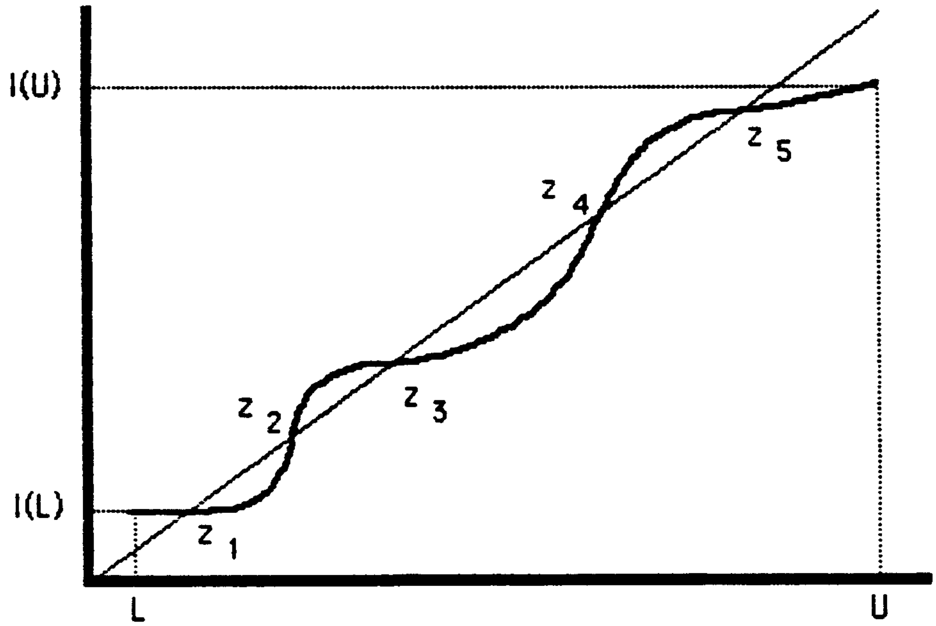


Figure 4. Illustrating Theorem 2.

The uniqueness of the solution for a network whose iteration function is known and satisfies the conditions of Theorem 2 is easy to test. Let z_L and z_U denote solutions obtained by starting with initial guesses $x_0 = L$ and $x_0 = U$, respectively. If $z_L = z_U$, the solution is unique. Otherwise the remaining solutions can be found by using the bisection method on the interval (z_L, z_U) and repeating the procedure in each subinterval.

The property expressed by Theorem 2 has been discussed by Courtois for the special case of throughput functions of multiprogrammed virtual memory systems [COUR75, COUR77]. It has also been used by Galler and Bos in their proof of convergence of their approximation for transaction blocking in database systems [GALL83].

4. CONVERGENCE PROOF EXAMPLES

This section will outline convergence proofs for three iteration schemata discussed in recent queueing network literature. The first example is the Bard-Schweitzer mean-value equations; this example falls within the scope of the first theorem. The second and third examples are the Jacobson-Lazowska surrogate server method and Sevcik's shadow CPU model; they fall within the scope of the second theorem.

4.1. Bard-Schweitzer MVA Approximation

The Bard-Schweitzer equations are an approximation to the equations of mean value analysis (MVA) for solving product-form queueing networks [BARD79, REIS80, SCHW79]. There are four equations for each $i = 1, \ldots, K$:

$$\bar{n}_i := \bar{n}_i^{\,*}$$

$$R_i := S_i \left(1 + \frac{N-1}{N} \bar{n}_i\right)$$

$$X_0 := N \;/\; \sum_{j=1}^{K} V_j R_j$$

$$\bar{n}_i^{\,*} := V_i R_i X_0$$

Starting with the initial condition $\bar{n}_i^{\,*} := N / K$, these $3K+1$ equations are evaluated repeatedly until all $| \bar{n}_i^{\,*} - \bar{n}_i | < \epsilon$. The symbols denote physical quantities as follows:

$\bar{n}_i$	=	mean queue length at device i
R_i	=	mean response time per visit to device i
S_i	=	mean service time at device i
V_i	=	visit ratio for device i
X_0	=	throughput of system
N	=	number of jobs in system
K	=	number of devices in system

These equations are an instance of the general model of Figure 1. The original model M_0 has $R_i = S_i\,(1+\bar{n}_{Ai})$, where $\bar{n}_{Ai}$ is the mean queue length seen by arrivals. According to the arrival theorem [BUZE80], the mean queue length seen by arrivals is the same as the overall mean queue length,, with the population reduced by 1 -- that is, $\bar{n}_{Ai}(N) = \bar{n}_i(N-1)$. A derived model, M, is used to determine an estimate of $\bar{n}_i(N-1)$ from $\bar{n}_i(N)$; Schweitzer chose $\frac{N-1}{N}\bar{n}_i(N)$ [BARD79]. The Mean Value Analysis equations result when the equation for model M is substituted into the equations for model M_0.

Until recently, no proof of convergence of this algorithm had been known. Then Eager-Sevcik and we independently showed that a unique solution is guaranteed for certain initial conditions. Eager and Sevcik assumed that initially all N jobs are enqueued at the bottleneck device [EAGE83]; their proof exploits the fact that the sequence of queue length estimates is monotonic. Their proof does not handle the initial condition stated in the algorithm given above. In their paper, de Sousa e Silva *et al.* showed the existence of a feasible solution for the multiclass version of this approximation but did not prove convergence [deSO83].

Appendix I contains a detailed convergence proof for the algorithm as stated. The idea is as follows. The iteration function of this algorithm is

$$\bar{n}_i = I(\bar{n}_1, \ldots, \bar{n}_K)$$

$$= \frac{ND_i\,(1+\frac{N-1}{N}\bar{n}_i)}{\sum_{j=1}^{K} D_j\,(1+\frac{N-1}{N}\bar{n}_j)} \qquad \text{for } i=1,\ldots,K$$

where $D_i = V_i S_i$. Without loss of generality, assume $D_1 < D_2 < \cdots < D_K$. Let $\bar{n}_i^{(0)} = N/K$, and $\bar{n}_i^{(1)}$, $\bar{n}_i^{(2)}$, ..., denote the sequence of mean queue length estimates at device i. Because this sequence is ultimately monotone and is also bounded (between 0 and N), Theorem 1 implies it converges. The same technique can be used for other initial conditions.

4.2. Jacobson-Lazowska Surrogate Server Method

Consider a system in which each job follows the behavior cycle:

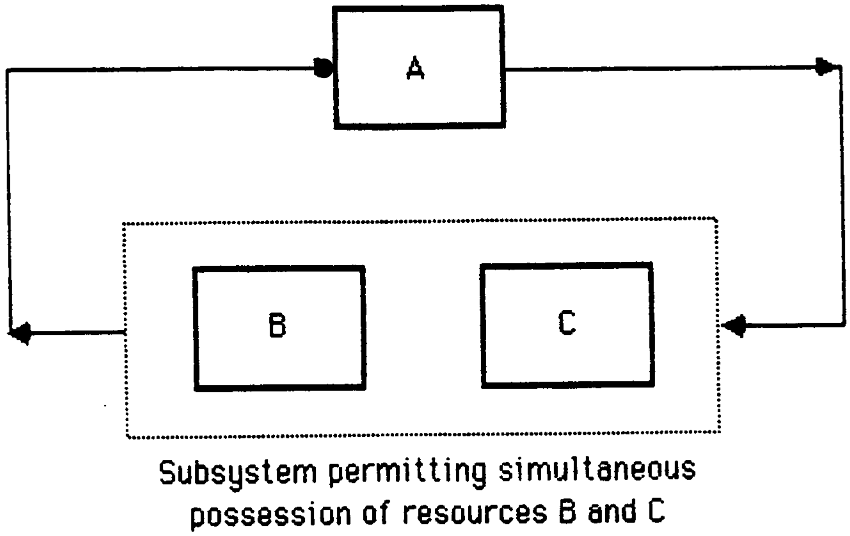
```
use A
request B
while holding B repeat until done
        {request C
        use B,C
        release C}
release B
```

(See Figure 5(a).) Ordinary queueing networks cannot model this case because they assume each job holds only one resource at a time.

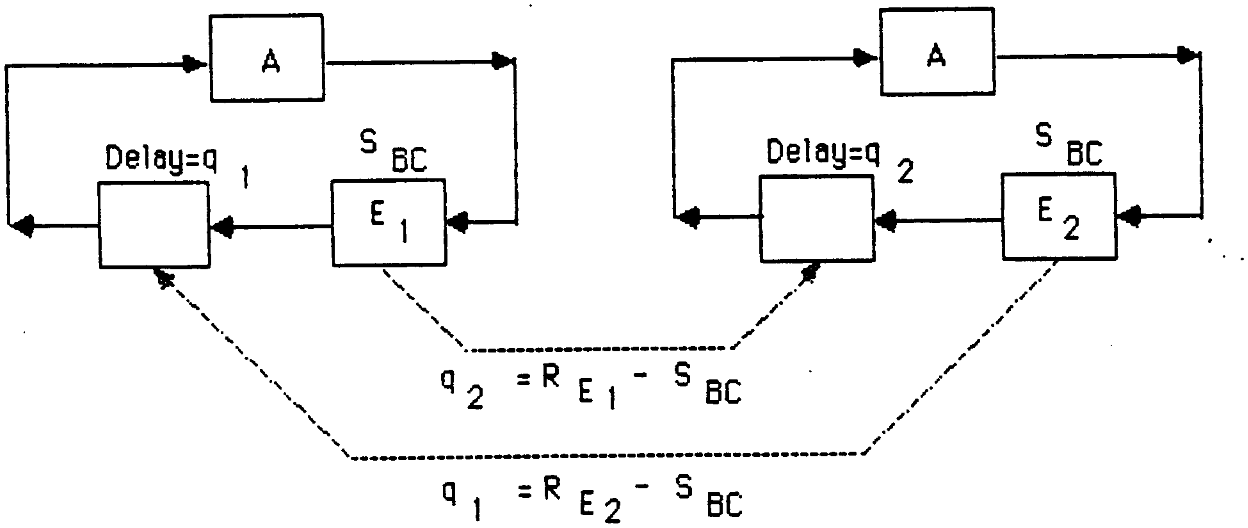
Jacobson and Lazowska analyzed this system with a pair of models (Figure 5(b)) [JACO82]. The idea is that model M_1 replaces the B-C subsystem with an equivalent server E_1 and a delay server; E_1 is flow-equivalent to the B-C subsystem with C removed and with service time S_{E_1} equal to S_{BC}, the sum of the B service and nonoverlapping C service. The delay server represents q_1, the delay caused by queueing for the C resource; q_1 is estimated by model M_2.

Model M_2 also contains an equivalent server E_2 and a delay server. Server E_2 is flow-equivalent to the B-C subsystem assuming no queueing occurs for B (i.e., assuming B is a pure service delay). The delay server represents q_2, the delay caused by queueing at the B resource; q_2 is esimated by model M_1.

Initially, the queueing delays q_1 and q_2 are unknown; they are determined iteratively by this algorithm:



(a) Model M_0



(b) Model M

FIGURE 5. Jacobson-Lazowska Surrogate Server Model.

```
set q_1* = 0
repeat {
      set q_1 = q_1*
      SOLVE M_1 for R_E1
      q_2 = R_E1 - S_BC = Y_2(q_1)
      SOLVE M_2 for R_E2
      q_1* = R_E2 - S_BC = Y_1(q_2) }
until | q_1* - q_1 | < ε .
```

Note that Y_1 and Y_2 are in general very complex functions of the model parameters. Nonetheless, only q_1 and q_2 change during the iteration. Therefore, for the purpose of analyzing the iteration, we can regard Y_1 and Y_2 as functions only of q_1 and q_2, respectively. Thus the iteration function is

$$I(q_1) = Y_1 . Y_2(q_1) .$$

Assuming that the service rates of A, B, and C are nondecreasing with respect to their queue lengths,

$$\frac{dY_2}{dq_1} < 0 \quad \text{and} \quad \frac{dY_1}{dq_2} < 0 ,$$

which implies

$$\frac{dI}{dq_1} = \frac{dY_1}{dq_2} \frac{dY_2}{dq_1} > 0 .$$

Since q_1 and q_2 are both nonnegative, and since q_1 is maximum when q_2 is zero, the Monotone Bounded Function Theorem (Theorem 2) implies that the algorithm will converge.

Jacobson and Lazowska gave a longer proof based on the principles of Theorem 1.

4.3. Sevcik's Shadow CPU Method

Product form queueing network models cannot represent servers that give priority to some job classes. Sevcik's method of replacing a CPU with two priority levels with two CPU's was discussed in Section 2 [SEVC77]. The iteration function is

$$U_H^{\,*} = \mathrm{SOLVE}(M, \{S_H, \frac{S_L}{1-U_H}\}) .$$

The derivative of this function is [AGRA83]

$$\frac{dU_H^{\,*}}{dU_H} = \frac{U_H}{1-U_H} [\, \bar{n}_L(N_L, N_H-1) - \bar{n}_L(N_L, N_H) \,] . \qquad (5)$$

This derivative is not, in general, less than 1 in magnitude everywhere. This is because both factors $U_H/(1-U_H)$ and $\bar{n}_L(N_L, N_H-1)-\bar{n}_L(N_L, N_H)$ may both be greater than 1. The second factor can be greater than 1 because adding a job to the H class can increase congestion elsewhere in the system and reduce the number of L jobs at the CPU. The best we can hope for is that the second factor is nonnegative, implying that the derivative is nonnegative and (by Theorem 2) at least one stable fixed point exists.

We conjecture that a sufficient condition for nonnegative derivative is that the rest of the network include only servers whose overall service-rate functions do not rise "superlinearly," i.e., they obey the constraint

$$\mu(n) \leqslant \frac{n}{n-1}\mu(n-1) . \qquad (6)$$

where $\mu(n)$ is the server's rate when the overall queue length is n. If a superlinear server is in the network, removal of a class H job may lead to much reduced service rate at that server. This will produce an increase in the class L transit time through the rest of the network, reduce the class L queue at the CPU, and make the derivative in Eq. (5) negative. The reasoning behind this statement is outlined in Appendix II.

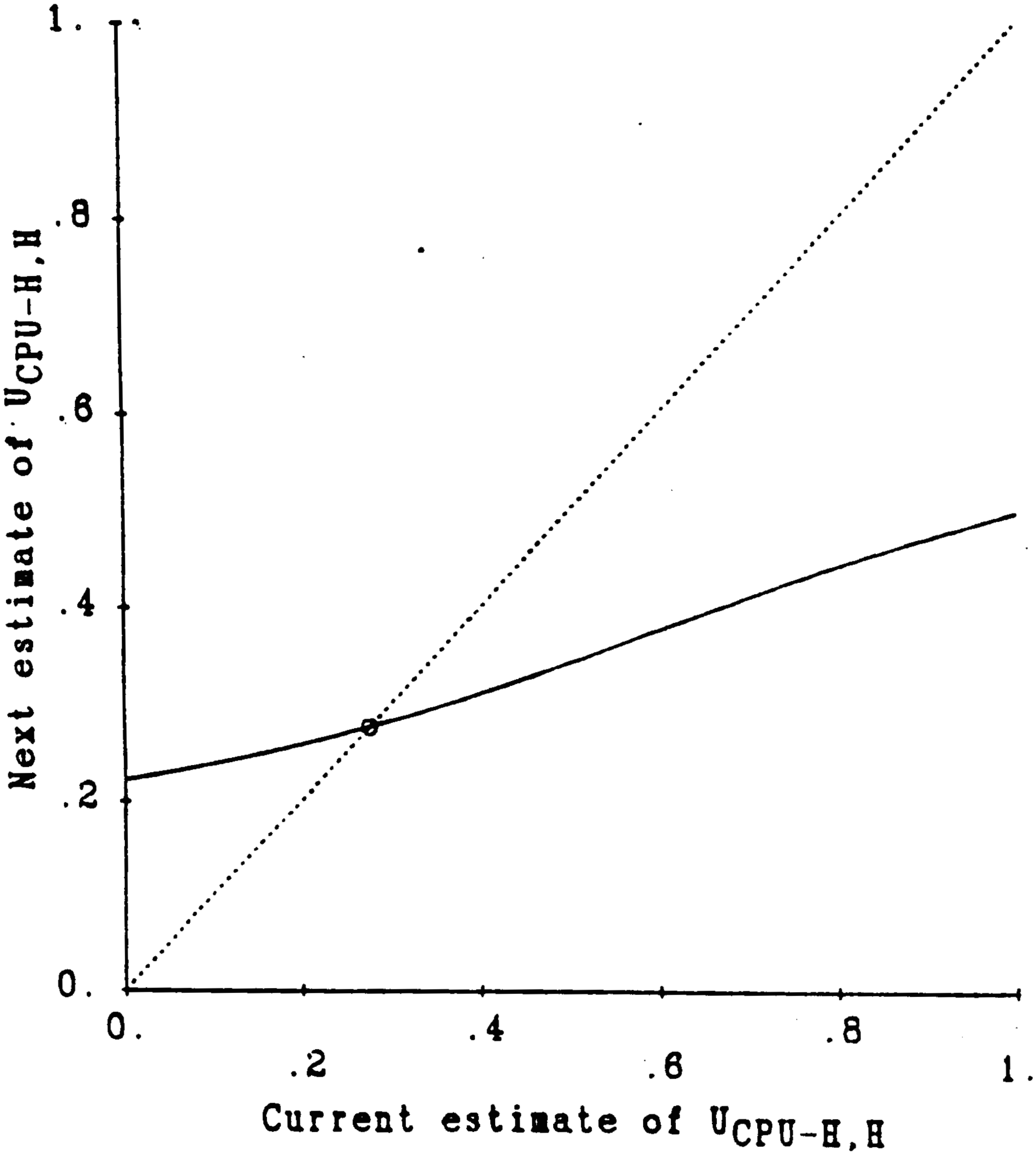
A superlinear server is, in effect, a "standby capacity" server that suddenly provides a sharp increase in resources when the queue length becomes sufficiently long. Such servers are not encountered in real systems. Real servers are typically fixed-rate servers, multiservers, delay servers, processor-sharing servers, servers whose rates increase less than linearly with queue length, and servers whose rates decrease with queue length. The generality of this class explains why the iteration function for the Sevcik model is monotone and why no one has found a practical system for which this model diverges.

Figure 6 shows a the parameters of a simple, two-station cyclic network with fixed rate servers. The Sevcik iteration function for this network has only one fixed point.

Figure 7 shows another network whose iteration has three fixed points; by Theorem 2, only two of them are stable. The iterative algorithm will converge on a solution that depends on the initial guess, $U_H^{(0)}$. For example, $U_H^{(0)}=0$ will cause convergence to the smaller solution, $U_H = 0.34$; $U_H^{(0)}=1$ will cause convergence to the larger solution, $U_H = 0.95$. Neither of these solutions is close to the value of $U_H = 0.66$ obtained by solving the exact model, the global balance equations. These two solutions, however, have a physical interpretation; we will report below on simulation results that indicate the system is bistable with two possible operating points.

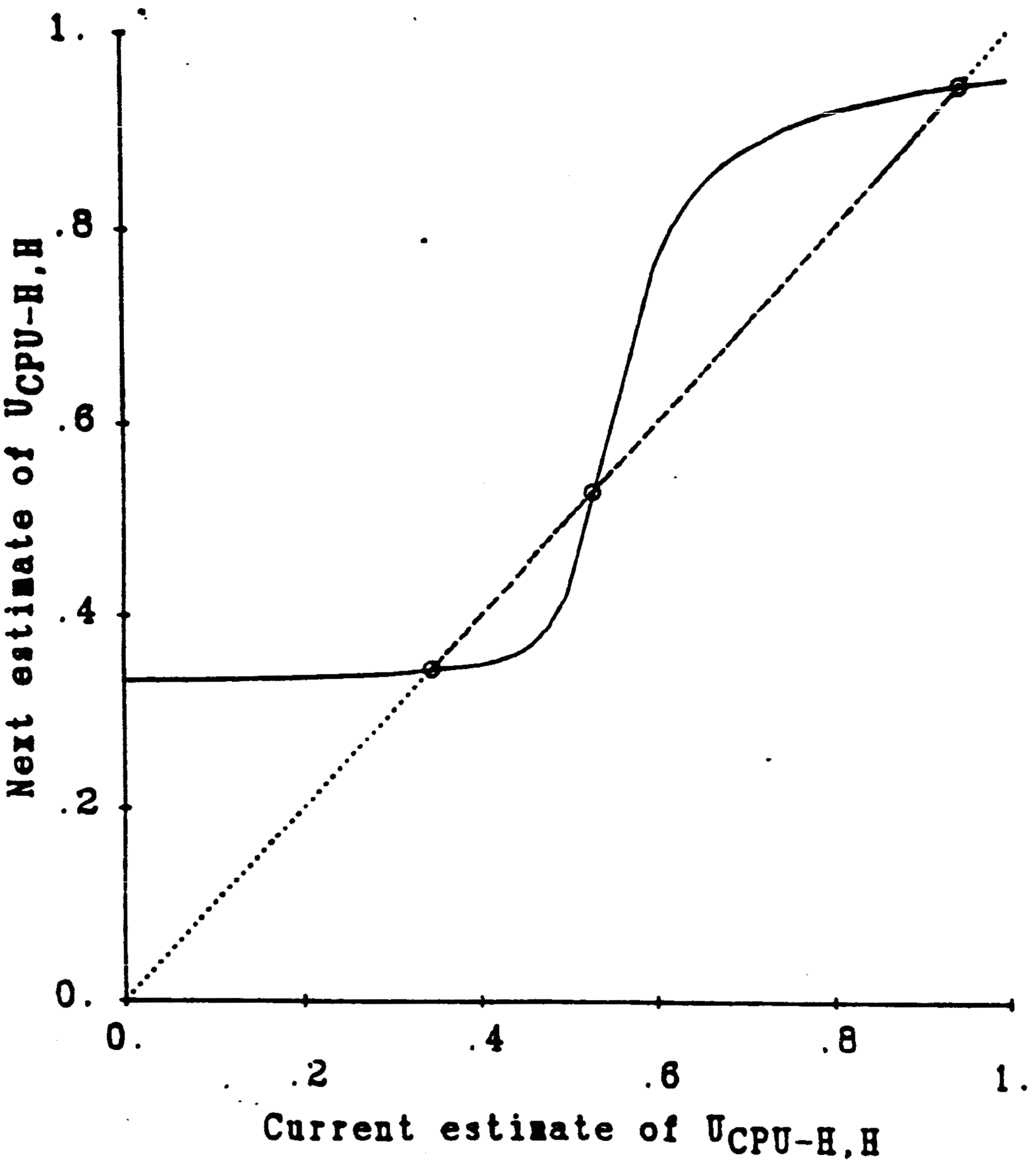
5. ANOMALIES

Two types of anomalous behavior can be encountered with iterative algorithms: divergence and multiple stable fixed points. These behaviors will be explained below and illustrated with examples in the Shadow CPU model.



NETWORK PARAMETERS			
CLASS	V_1S_1	V_2S_2	N
H	1.0	1.0	1
L	1.0	1.0	5

FIGURE 6. Two-station Shadow CPU network with one stable fixed point.



NETWORK PARAMETERS					
CLASS	V_1	S_1	V_2	S_2	N
H	1	0.1	1	0.005	1
L	1	0.05	21	0.005	40

FIGURE 7. Two-station Shadow CPU Network with two stable fixed points.

5.1. Divergence

If the derivative of the iteration function is negative in some range, the function is not monotone increasing and Theorem 2 may not apply. An iterative algorithm may fail to find any solution for such a network. As noted above, such a network must contain a superlinear server.

Figure 8 shows the parameters of a two-station cyclic network including a superlinear server with rate function

$$\{\mu(n),\ n=1,\ldots,6\} \quad = \quad \{1,1,1,1,1,1000\} .$$

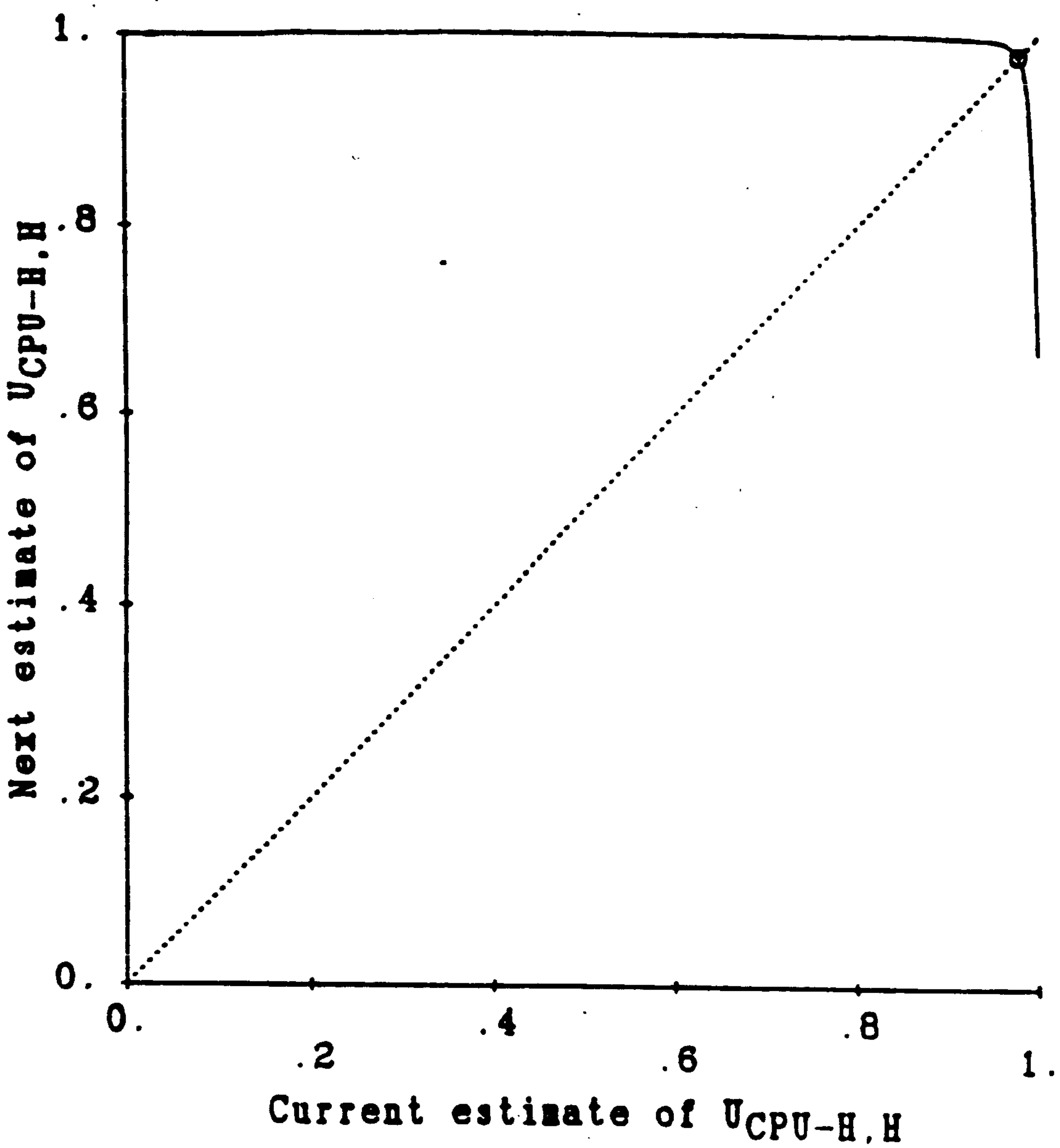
The iteration function has a negative slope everywhere. The function has one unstable fixed point. Hence the iterative algorithm will suffer oscillatory divergence.

5.2. Multistable Fixed Points

Figure 7 showed an instance of a network having two stable solutions; the iterative algorithm can find either one depending on the initial guess of U_H. Do these fixed points represent physically observable phenomena, or are they a defect of the modeling technique?

To answer this question, we simulated the network of Figure 7 for 10,000 simulated seconds. A trace of the class H CPU utilization (U_H) over each 5-second interval is displayed in Figure 9. (Class H CPU service time per visit, S_H, is 0.1 seconds and Class L CPU service time per visit, S_L, is 0.05 second.) While U_H is normally around 95%, the system occasionally enters the state in which U_H is low (about 35%) for a significant period. We conclude that the two stable values of U_H predicted by the model are both stable operating points in the corresponding real system.

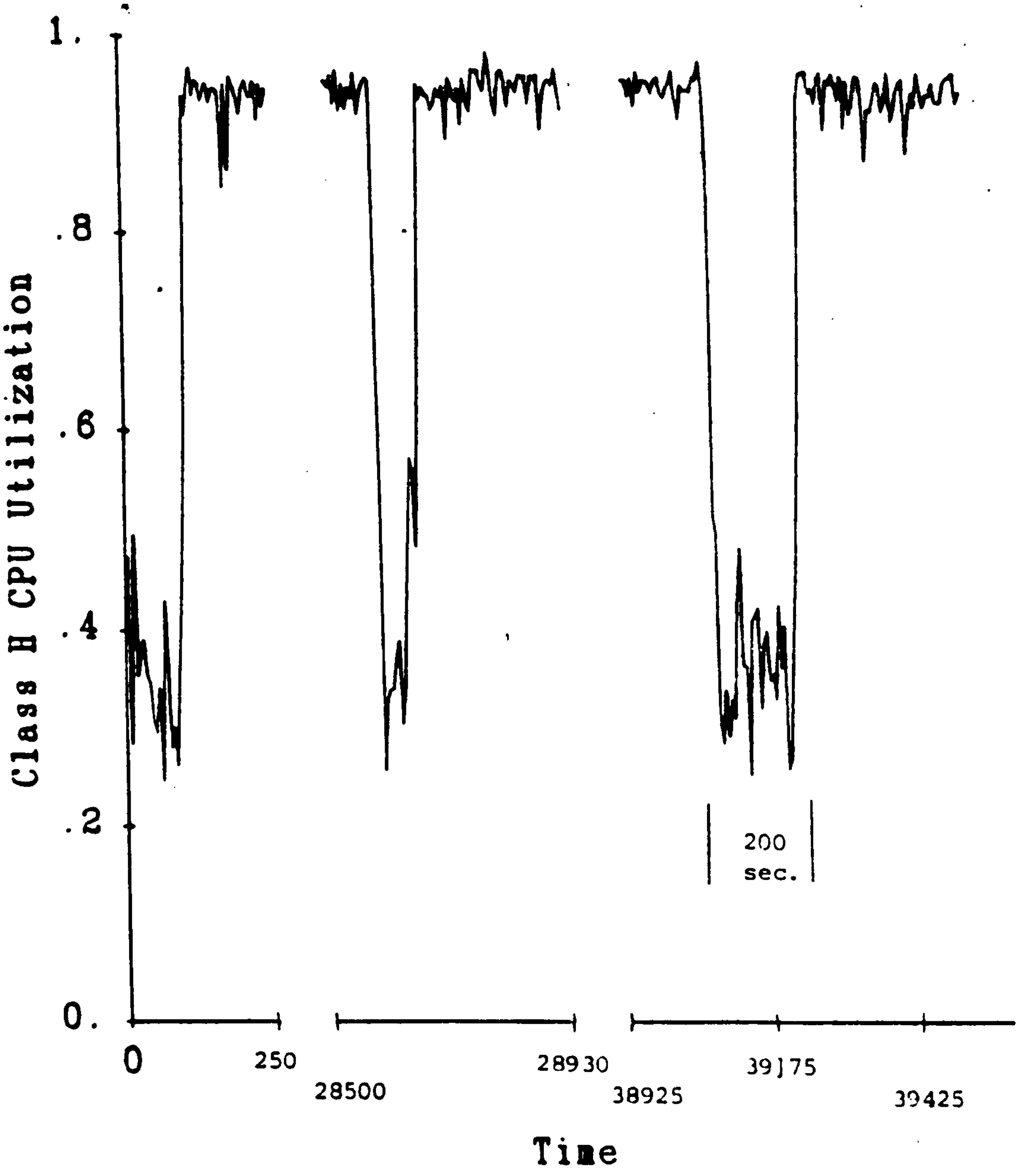
Figure 10 shows the probability distribution for the network. Curve C is the marginal CPU queue length distribution for Class L. It has two well-defined peaks. The right peak occurs when almost all Class L jobs are queued at the CPU; in these states the Class H job experiences



NETWORK PARAMETERS			
CLASS	V_1S_1	V_2S_2	N
H	10.0	10.0	2
L	1.0	1000.0	5

$Rate_2(n) = \{1, 1, 1, 1, 1, 1000\}$

FIGURE 8. Two-station Shadow CPU network with no stable fixed point.



Each sample indicates CPU utilization in the preceding 5 second interval. CPU utilization during the intervals 250 - 28500 and 28930 - 38925 seconds was about 95%.

FIGURE 9. Simulation trace of Class H CPU utilization for the model of Figure 7, showing both utilizations occurring in different intervals.

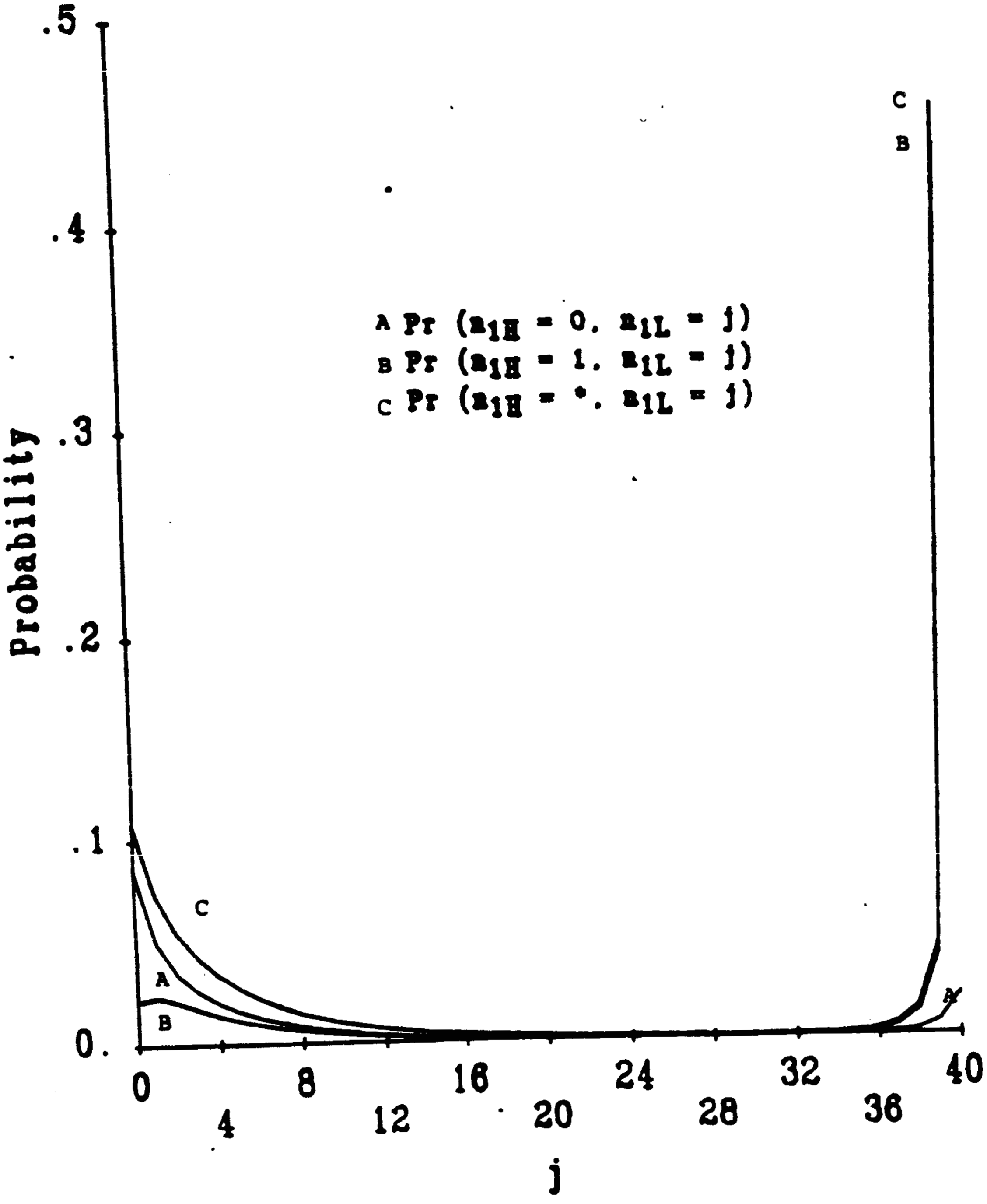


FIGURE 10. Probability distribution of observed Class H utilization for the model of Figure 7.

little contention at Server 2 allowing both throughput X_H and utilization U_H to be high. This fact is demonstrated by the dominance of the conditional probability distribution curve B $(P(N_{1L} \mid N_{1H}=1))$ over the conditional probability curve A $(P(N_{1L} \mid N_{1H}=0))$. The opposite behavior occurs at the left peak when almost all Class L jobs are queued at Server 2. In this case. the Class H job experiences significant delay at Server 2 because priorities are not honored there, forcing both X_H and U_H to be low. This argument is supported by the dominance of curve A over curve B on the left side of the picture. The relatively higher right peak corresponds to the observation from the simulation that the higher class H CPU utilization is preferred.

These bistable results are reminiscent of results observed by Courtois for multiprogrammed virtual memory systems susceptible to thrashing [COUR75, COUR77]. In the high-throughput (CPU utilization) state, jobs spend little time waiting at paging devices. In the thrashing state, the page wait time increases sharply and throughput decreases. Figure 11 helps visualize the problem. The horizontal axis is n, the number of thinking terminals. The straight line $\lambda(N-n)$ denotes the arrival rate of work to the central subsystem when n jobs are active ($N-n$ jobs are thinking) and the think time is $1/\lambda$. The curve $\mu(n)$ denotes the output rate of the central subsystem. The equation

$$\lambda(N-n) = \mu(n)$$

expresses a form of flow balance. For certain choices of the parameter λ there will be three fixed points (compare with Figure 7). The down-crossings, z_1 and z_3 are stable while the up-crossing z_2 is unstable. The system will have a high probability of being observed with n near z_1 or z_3 and a low probability of being observed with n near z_2.

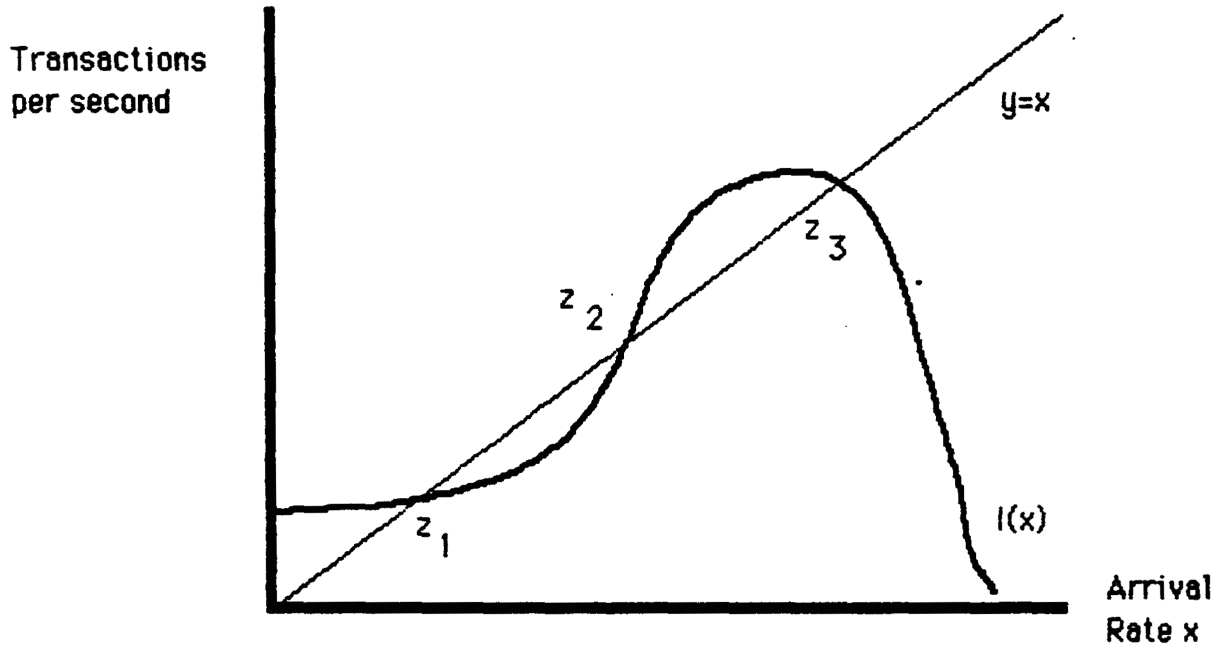


FIGURE 11. Application of stability theorem to virtual memory system. The straight line (y=x) is the arrival rate to the virtual memory subsystem and the iteration function I(x) is the corresponding output rate.

6. CONCLUSIONS

Iteration arises in the solution of queueing network models when the parameters of a derived model depend on unknown performance metrics of the original model. Iteration can be used to refine successive guesses of the unknown metrics until mutually consistent values of metrics and parameters are found.

There are two basic approaches to establishing whether an iterative model converges to a solution. One is to construct the iteration function and demonstrate that the sequence of estimates is bounded and ultimately monotonic. The other is to show that the iteration function is itself bounded and monotonic, in which case iterative solutions are guaranteed to converge. We applied these methods to prove the convergence of the Bard-Schweitzer approximate MVA equations, the Jacobson-Lazowska surrogate server model, and Sevcik's shadow CPU model.

We also showed that, in general, a model with monotone iteration function may have more than one convergent solution. Which one is obtained depends on the initial conditions. We used the shadow CPU model to illustate that multiple stable solutions exist and have physical interpretations.

We argued that the shadow CPU model can be divergent for all initial conditions if at least one non-CPU server is sufficiently superlinear -- i.e., becomes increasingly fast as the load on it increases. Because such servers are not implemented in practice, we concluded that solutions for this model applied to practical systems are always convergent. We speculate that this property holds for any queueing network: real servers, whose service functions are typically concave downward in the queue length, always generate monotone iteration functions.

7. ACKNOWLEDGEMENT

We deeply appreciate comments from Ken Sevcik, which led to an improved and more complete paper.

8. BIBLIOGRAPHY

AGRA83 S. C. Agrawal, "Metamodeling: a study of approximations in queueing models," PhD Thesis, Computer Sciences Department, Purdue University, West Lafayette, IN 47907, August 1983.

BARD79 Y. Bard, "Some extensions to multiclass queueing network analysis," in *Performance of Computer Systems* (Arato et al., Eds.), North-Holland, New York (1979), 51-62.

BUZE73 J. P. Buzen, "Computational algorithms for closed queueing networks with exponential servers," *Communications ACM 16*, 9 (September 1973), 527-531.

BUZE80 J. P. Buzen and P. J. Denning, "Measuring and calculating queue length distributions," IEEE *Computer 13*, 4 (April 1980), 33-44.

COUR75 P. J. Courtois, "Decomposability, instabilities, and saturation in multiprogramming systems," *Communications ACM 18*, 7 (July 1975), 371-376.

COUR77 P. J. Courtois, *Decomposability: Queueing and Computer System Applications*, Academic Press, New York (1977).

deSO83 E. de Sousa e Silva, S. S. Lavenberg, and R. R. Muntz, "A perspective on iterative methods for the approximate analysis of queueing networks," *Proc. Int'l Workshop on Applied Mathematics and Performance and Reliability of Computer and Communication Systems* (G. Iazeolla and S. Tucci, Eds.), North-Holland Publishing Co. (1983).

DENN78 P. J. Denning and J. P. Buzen, "The operational analysis of queueing network models," *Computing Surveys 10*, 3 (September 1978), 225-261.

EAGE83 D. L. Eager and K. C. Sevcik, "An analysis of an approximation algorithm for queueing networks," Department Computer Science, University of Toronto (1983), submitted to ACM TOCS.

GALL83 B. I. Galler and L. Bos, "A model of transaction blocking in databases," *Performance Evaluation 3*, 2 (May 1983), 95-122.

GORD80 K. D. Gordon and L. W. Dowdy, "The impact of certain parameter estimation errors in queueing network models, *Proc. Performance '80*, printed in *Performance Evaluation Review* (Summer 1980), 3-9.

JACO82 P. A. Jacobson and E. D. Lazowska, "Analyzing queueing networks with simultaneous resource possession," *Communications ACM 25*, 2 (February 1982), 142-151.

REIS80 M. Reiser and S. S. Lavenberg, "Mean value analysis of closed multichain queueing networks," *J. ACM 27*, 2 (April 1980), 313-322.

SCHW79 P. Schweitzer, "Approximate analysis of multiclass closed networks of queues," *Int'l Conf. on Stochastic Control and Optimization*, Amsterdam, Netherlands (1979).

SEVC77 K. C. Sevcik, "Priority scheduling disciplines in queueing network models of computer systems," *Proc. IFIP Congress 77*, North-Holland, Amsterdam (1977), 565-570.

STOE80 J. Stoer and R. Bulrisch, *Introduction to Numerical Analysis*, Springer-Verlag, New York (1980).

SURI83 R. Suri, "Characterization of monotonicity in closed queueing networks," Report TR-06-83, Center for Research in Computing Technology, Harvard University, Cambridge, MA 02138 (February 1983).

WILL76 A. C. Williams and R. A. Bhandiwad, "A generating function approach to queueing network models of multiprogrammed computer systems," *Network 6* (1976), 1-22.

9. APPENDIX I -- Proof of Convergence of Bard-Schweitzer Approximation

Let $\bar{n}_i^{(m)}$ denote the m^{th} estimate of the queue length $\bar{n}_i$ obtained from the Bard-Schweitzer iteration function

$$\bar{n}_i^{(m)} = \frac{ND_i\,(1+\frac{N-1}{N}\bar{n}_i^{(m-1)})}{\sum_{j=1}^{K} D_j\,(1+\frac{N-1}{N}\bar{n}_j^{(m-1)})} \quad \text{for } i=1,\ldots,K \text{ and } m>0 .$$

If we can show that this sequence is ultimately monotonic, we can apply Theorem 1 to deduce its convergence. We will prove a more detailed proposition,from which the desired result follows.

We will use the following notation. Let $\bar{\mathbf{n}}^{(m)} = (\bar{n}_1^{(m)}, \ldots, \bar{n}_K^{(m)})$ denote the vector of mean queue length estimates obtained after the m^{th} iteration. Let $\mathbf{D} = (D_1, \ldots, D_K)$ denote the vector of total service demands $(D_i = V_i S_i)$ for each of the devices; without loss of generality, we suppose that $D_1 < \cdots < D_K$.

Proposition: For all $m>0$ there exists a device index p_m (the "partition index") satisfying:

(a). $i \leqslant p_m \;=>\; n_i^{(m)} < n_i^{(m-1)}$

(b). $i > p_m \;=>\; n_i^{(m)} > n_i^{(m-1)}$

(c). $p_m \geqslant p_{m-1}$ (take $p_0 = 0$)

(d). $\bar{n}_1^{(m)} < \cdots < \bar{n}_K^{(m)}$.

As m increases, there will be an ultimate value of p_m in the set $\{1,\ldots,K\}$. These statements say that all the devices numbered $1,\ldots,p_m$ have ultimately monotonic decreasing queue length estimates and moreover that the values of the estimates are ordered the same as the values of the total demand.

Proof:

Basis ($m=1$): Substituting $\bar{n}_i^{(0)} = N/K$ into the iteration formula,

$$\bar{n}_i^{(1)} = \frac{ND_i(1+\frac{N-1}{N}\frac{N}{K})}{\sum_{j=1}^{K} D_j(1+\frac{N-1}{N}\frac{N}{K})} = N\frac{D_i}{\sum_{j=1}^{K} D_j}.$$

Thus the ordering of $\{D_i\}$ implies the same ordering on the $\{\bar{n}_i^{(1)}\}$ and Part d is true. Because the estimates are now unequal and ordered, some of them must be less than N/K. So let p_1 be the largest device index such that $\bar{n}_i^{(1)} < N/K$. Now:

$$i \leqslant p_1 \ \Rightarrow\ \bar{n}_i^{(1)} < \bar{n}_1^{(0)},$$

$$i > p_1 \ \Rightarrow\ \bar{n}_i^{(1)} > \bar{n}_1^{(0)}, \text{ and}$$

$$p_1 > p_0 = 0.$$

which establishes Parts a-c.

Induction: (1). We note first that $\bar{\mathbf{n}}^{(m)}\bullet\mathbf{D} > \bar{\mathbf{n}}^{(m-1)}\bullet\mathbf{D}$. This is because (by hypothesis) the transformation from $m-1$ to m reduces all the $\bar{n}_i^{(m-1)}$ for $i \leqslant p_m$ and adds the total reduction to the $\bar{n}_i^{(m-1)}$ for $i > p_m$, thereby shifting value from terms of less weight to terms of greater weight.

(2). Next we note that $i \leqslant p_m$ implies $\bar{n}_i^{(m+1)} < \bar{n}_i^{(m)}$ -- i.e., the monotonic decrease continues. To see this note

$$\bar{n}_i^{(m+1)} = \frac{ND_i(1+\frac{N-1}{N}\bar{n}_i^{(m)})}{\mathbf{D}\bullet 1+\frac{N-1}{N}\mathbf{D}\bullet\bar{\mathbf{n}}^{(m)}} < \frac{ND_i(1+\frac{N-1}{N}\bar{n}_i^{(m-1)})}{\mathbf{D}\bullet 1+\frac{N-1}{N}\mathbf{D}\bullet\bar{\mathbf{n}}^{(m-1)}} = \bar{n}_i^{(m)},$$

where the "<" in the numerator follows from Part a of the induction hypothesis and the ">" in the denominator follows from the observation (1) above.

(3). The induction hypothesis implies $D_i < D_j \;\Rightarrow\; \bar{n}_i^{(m)} < \bar{n}_j^{(m)}$. Now, $D_i < D_j$ implies

$$\bar{n}_i^{(m+1)} = \frac{ND_i\,(1+\frac{N-1}{N}\bar{n}_i^{(m)})}{\mathbf{D}\bullet 1+\frac{N-1}{N}\mathbf{D}\bullet\bar{\mathbf{n}}^{(m)}} < \frac{ND_j\,(1+\frac{N-1}{N}\bar{n}_j^{(m)})}{\mathbf{D}\bullet 1+\frac{N-1}{N}\mathbf{D}\bullet\bar{\mathbf{n}}^{(m)}} = \bar{n}_j^{(m+1)},$$

which establishes Part d.

(4). Next we define p_{m+1} to be the largest device index $i \geqslant p_m$ for which $\bar{n}_i^{(m+1)} < \bar{n}_i^{(m)}$. This means that $p_m < i \leqslant p_{m+1}$ implies that the device-i mean queue estimate switched from increasing to decreasing at the m^{th} iteration. Note p_{m+1} cannot be smaller than p_m because by (1) a queue-length estimate continues to decrease. (Note that if p_m has reached its final value, all $\bar{n}_i^{(m)}$ for $i \geqslant p_m$ will be monotonically increasing for all larger m.) We may now conclude:

$$i \leqslant p_{m+1} \;\Rightarrow\; \bar{n}_i^{(m+1)} < \bar{n}_i^{(m)};$$

$$i > p_{m+1} \;\Rightarrow\; \bar{n}_i^{(m+1)} > \bar{n}_i^{(m)}; \text{ and}$$

$$p_{m+1} \geqslant p_m;$$

as was to be shown. Now the convergence of the Bard-Schweitzer estimator follows from Theorem 1.

Corollary: Let $R_0^{(m)}$ denote the system mean response time estimate after the m^{th} iteration. The sequence $\{R_0^{(m)} \mid m \geqslant 0\}$ is monotonic increasing.

Proof: By definition,

$$R_0^{(m)} = \sum_{j=1}^{K} V_j R_j^{(m)} = \mathbf{D}\bullet 1+\frac{N-1}{N}\mathbf{D}\bullet\bar{\mathbf{n}}^{(m)}.$$

But we showed that the second term in this sum is monotone increasing, which implies that the entire sum is monotone increasing.

10. APPENDIX II -- Superlinear Servers

We are interested in the class of networks for which the second term in Eq. (5) is nonnegative. Consider the queue length at CPU-L:

$$\begin{aligned} \bar{n}_L(N_L, N_H) &= X_L R_L \\ &= X_L S_L'(1+\bar{n}_L(N_L-1, N_H)) \\ &= S_L' \sum_{i=1}^{N_L} \prod_{j=i}^{N_L} X_L(N_L-j, N_H) \end{aligned} \qquad \text{(AII.1)}$$

where S_L' is the class L service time at CPU-L. From (AII.1) it is clear that $\bar{n}_L(N_L, N_H-1) \geqslant \bar{n}_L(N_L, N_H)$ whenever

$$X_L(n_L, N_H-1) \geqslant X_L(n_L, N_H), \quad n_L = 1, \ldots, N_L \ . \qquad \text{(AII.2)}$$

Intuitively, Eq. AII.2 will hold if the processing rate (μ) for each class at each server in the transformed network satisfies the pair of conditions

$$\mu_L(n_L, n_H-1) \geqslant \mu_L(n_L, n_H)$$
$$\mu_H(n_L, n_H-1) \geqslant \mu_H(n_L, n_H)$$

In a product form network, the service rate μ of a load dependent server can depend only on the total local queue length n $(=n_L+n_H)$. Therefore,

$$\mu_L(n_L,n_H) = \frac{n_L}{n_L+n_H}\frac{\mu(n_L+n_H)}{S_L} = \frac{n_L}{n}\frac{\mu(n)}{S_L}$$

$$\mu_H(n_L,n_H) = \frac{n_H}{n_L+n_H}\frac{\mu(n_L+n_H)}{S_H} = \frac{n_H}{n}\frac{\mu(n)}{S_H} \qquad \text{(AII.3)}$$

On substituting (AII.3) into (AII.2), we obtain the sufficient constraint for the monotonicity of Eq. (5) of the text:

$$\mu(n) \leqslant \frac{n}{n-1}\mu(n-1)$$

This constraint is equivalent to the requirement that $nS(n)$, where $S(n)$ is the overall service function of the server, is nondecreasing in n. It is interesting to observe that this condition is precisely the constraint on the service function of the central subsystem employed by Galler and Bos in their proof of convergence of a different system [GALL83]. It is also interesting that the overall mean response time per visit to server i can be written

$$R_i(N) = \sum_{n=1}^{N} nS(n)\, p_{Ai}(n-1,N),$$

where p_{Ai} is the arrival distribution. The increasing function $nS(n)$ implies that all the mean response times, cumulative queue length distributions, throughputs, and utilizations are increasing with respect to load N. (See also [SURI83.]